IN THE CLAIMS

Please cancel Claim 3, without prejudice.

Please amend Claim 4, as follows.

A8

1. (Original) A method for use by a host microprocessor which translates sequences of instructions from a target instruction set for a target processor to sequences of instructions for the host microprocessor comprising the steps of:

beginning execution of a first sequence of target instructions by committing state of the target processor and storing memory stores generated by previously-executed sequences of instructions at a point in the execution of instructions at which state of the target processor is known,

beginning execution of a speculative sequence of host instructions following a branch from the first sequence of target instructions by immediately committing state and storing memory stores,

attempting to execute the speculative sequence of host instructions until another point in the execution of target instructions at which state of the target processor is known,

rolling back to last committed state of the target processor and discarding memory stores generated by the speculative sequence of host instructions if execution fails, and

beginning execution of a next sequence of target instructions if execution succeeds.

A8

Claim 2. (Original) A method as claimed in Claim 1 including an additional step of releasing a lock for any sequence of host instructions running in a locked condition immediately after committing state of the target processor and storing memory stores generated by previously-executed translation sequences.

Claim 3. (Cancelled, without prejudice)

Claim 4. (Currently Amended) A method for use by a host microprocessor which translates sequences of instructions from a target instruction set for a target processor to sequences of instructions for the host microprocessor comprising the steps of:

translating a first speculative sequence of host instructions from ~~the~~ a first sequence of target instructions from a point in the translation of target instructions at which state of the target processor is known,

ending the first sequence of target instructions in response to encountering a branch from the first sequence in the target program by:

branching to a branch sequence of target instructions,
committing state of the target processor and storing memory stores generated by the first translation sequence after a branch taken before executing a branch sequence of host instructions, and

ending execution of the first sequence of target instructions if a branch is not taken from the first sequence by:

A8

rolling back to last committed state of the target processor and discarding memory stores generated by the speculative sequence of host instructions if execution fails, and committing state of the target processor and storing memory stores generated by the first sequence at the end of the sequence of target instructions at which state of the target processor is known.